*献辞*

*To Jordan, for her unwavering support no matter*

*how many times she heard “it’s almost done.”*

马特弗里斯比

ABOUT THE AUTHOR

MATT FRISBIE has worked in web development for over a decade. During that time, he’s been a

startup co-founder, an engineer at a Big Four tech company(四大科技公司之一), and the first engineer at a Y Combinator startup that would eventually become a billion-dollar business. As a Google software engineer, Matt worked on both the AdSense and Accelerated Mobile Pages (AMP) platforms; his code contributions run on most of the planet’s web browsing devices.

Prior to this, Matt was the first engineer at DoorDash, where he laid the foundation for their driver scheduling, menu management, and order dispatch infrastructure. Matt has written two books and recorded two video series for O’Reilly and Packt, speaks at frontend meetups and webcasts, and is a Level 1 sommelier. He majored in computer engineering(计算机科学) at the University of Illinois(伊利诺斯州（美国州名）) at Urbana–Champaign(伊利诺伊大学香槟分校). Matt’s Twitter handle is @mattfriz.

**CHAPTER 6 Collection Reference Types**

Collection ：收藏品，物；

6.1 THE OBJECT TYPE

Although instances of Object don’t have much functionality(功能), they are ideally suited to storing and transmitting data around an application(应用程序).

There are two ways to explicitly create an instance of Object:

* The first is to use the new operator with the Object constructor
* The other way is to use object literal notation.

In this example, the left curly brace ({) signifies the beginning of an object literal because it occurs in an expression context(表达式上下文).

CHAPTER 8 Objects, Classes, and Object-Oriented Programming

it helps to think of ECMAScript objects as hash tables(哈希表): nothing more than a grouping of namevalue pairs where the value may be data or a function

8.1 UNDERSTANDING OBJECTS

The canonical([典型](javascript:;)) way of creating a custom object is to create a new instance of Object and add properties and methods to it, as in this example:

主语 谓语 宾语

let person = new Object();

person.name = "Nicholas";

person.age = 29;

person.job = "Software Engineer";

person.sayName = function() {

console.log(this.name);

};

The sayName() method displays the value of this.name, which resolves to person.name.

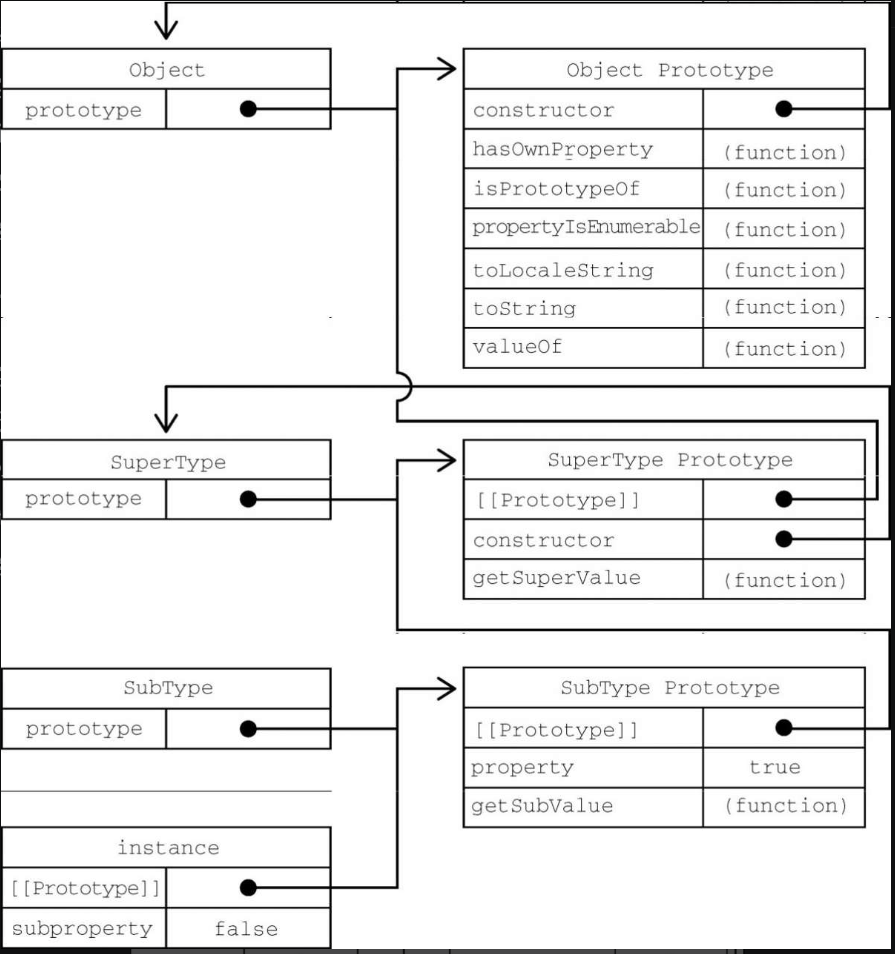
8.3 继承

总述：

javascript中的面向对象编程有关继承的实现方式只有“实现继承”，这种继承能继承实际的方法。实现继承主要通过原型链实现。原型链的基本构想是在实例和原型之间构造了一条原型链，

8.3.1 原型链

1．默认原型与图8-5展示了完整的原型链



2．原型与继承关系：确定原型与实例的关系的两种方式

原型搜索机制:

文脉：叙述过程，思考逻辑

3．关于方法：子类有时候需要覆盖父类的方法

原型链的问题

主要问题：原型包含引用值导致的继承问题。

属性通常会在构造函数中定义而不会定义在原型上的原因

“盗用构造函数”

function SuperType() {

this.colors = ["red", "blue", "green"];

}

function SubType() {

//继承SuperType

SuperType.call(this);

}

let instance1 = new SubType();

instance1.colors.push("black");

console.log(instance1.colors); // "red, blue, green, black"

let instance2 = new SubType();

console.log(instance2.colors); // "red, blue, green"

8.3.2 盗用构造函数

扩展

[ECMAScript® 2022 Language Specification (ecma-international.org)](https://262.ecma-international.org/13.0/" \l "sec-terms-and-definitions-prototype)

官网

Scope：适用范围

# 4.4.8 prototype

object that provides shared properties for other objects

注释：

SubType.prototype = new SuperType();  
所谓的new一个构造函数，可以理解为给prototype对象添加属性和方法

NOTE

When a [constructor](https://262.ecma-international.org/13.0/" \l "constructor) creates an object, that object implicitly references the [constructor](https://262.ecma-international.org/13.0/" \l "constructor)'s "prototype" property for the purpose of resolving property references. The [constructor](https://262.ecma-international.org/13.0/" \l "constructor)'s "prototype" property can be referenced by the program expression **constructor.prototype**, and properties added to an object's prototype are shared, through inheritance, by all objects sharing the prototype. Alternatively, a new object may be created with an explicitly(显式地) specified prototype by using the **Object.create** built-in function.

# 20.2.4.3 prototype

Function instances that can be used as a [constructor](https://262.ecma-international.org/13.0/" \l "constructor) have a "prototype" property. Whenever such a Function instance is created another [ordinary object](https://262.ecma-international.org/13.0/" \l "ordinary-object) is also created and is the initial value of the function's "prototype" property. Unless otherwise specified, the value of the "prototype" property is used to initialize the [[Prototype]] internal slot of the object created when that function is invoked as a [constructor](https://262.ecma-international.org/13.0/" \l "constructor).

This property has the attributes { [[Writable]]: true, [[Enumerable]]: false, [[Configurable]]: false }.

NOTE

[Function objects](https://262.ecma-international.org/13.0/" \l "function-object) created using **Function.prototype.bind**, or by evaluating a [MethodDefinition](https://262.ecma-international.org/13.0/" \l "prod-MethodDefinition) (that is not a [GeneratorMethod](https://262.ecma-international.org/13.0/" \l "prod-GeneratorMethod) or [AsyncGeneratorMethod](https://262.ecma-international.org/13.0/" \l "prod-AsyncGeneratorMethod)) or an [ArrowFunction](https://262.ecma-international.org/13.0/" \l "prod-ArrowFunction) do not have a "prototype" property.

# 27.3.4.3 prototype

Whenever a GeneratorFunction instance is created another [ordinary object](https://262.ecma-international.org/13.0/" \l "ordinary-object) is also created and is the initial value of the generator function's "prototype" property. The value of the prototype property is used to initialize the [[Prototype]] internal slot of a newly created Generator when the generator [function object](https://262.ecma-international.org/13.0/" \l "function-object) is invoked using [[Call]].

This property has the attributes { [[Writable]]: true, [[Enumerable]]: false, [[Configurable]]: false }.

NOTE

Unlike Function instances, the object that is the value of the a GeneratorFunction's "prototype" property does not have a "constructor" property whose value is the GeneratorFunction instance.

# 27.4.4.3 prototype

Whenever an AsyncGeneratorFunction instance is created another [ordinary object](https://262.ecma-international.org/13.0/" \l "ordinary-object) is also created and is the initial value of the async generator function's "prototype" property. The value of the prototype property is used to initialize the [[Prototype]] internal slot of a newly created AsyncGenerator when the generator [function object](https://262.ecma-international.org/13.0/" \l "function-object) is invoked using [[Call]].

This property has the attributes { [[Writable]]: true, [[Enumerable]]: false, [[Configurable]]: false }.

NOTE

Unlike function instances, the object that is the value of the an AsyncGeneratorFunction's "prototype" property does not have a "constructor" property whose value is the AsyncGeneratorFunction instance.

# 23.1.2.4 Array.prototype

The value of **Array.prototype** is the [Array prototype object](https://262.ecma-international.org/13.0/" \l "sec-properties-of-the-array-prototype-object).

This property has the attributes { [[Writable]]: false, [[Enumerable]]: false, [[Configurable]]: false }.